Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»
Курс «Базовые компоненты интернет-технологий»

Отчёт по Рубежному Контролю №1
по курсу «Базовые компоненты и интернет-технологии»
Вариант 16

Выполнил:
студент группы ИУ5-33Б
Рыбин Владислав

Проверил:
к.т.н., доц., Ю. Е. Гапанюк

2022 г.

Полученное задание:

Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

Необходимо разработать запросы в соответствии с Вашим вариантом.

Предметная область: класс_1 – Книга, класс_2 – Книжный магазин, вариант запросов: В.

Запросы:

1. «Книга» и «Книжный магазин» связаны соотношением один-ко-многим. Выведите список всех книжных магазинов, у которых название начинается с буквы «А», и их книги.
2. «Книга» и «Книжный магазин» связаны соотношением один-ко-многим. Выведите список магазинов с минимальной ценой книги в каждом, отсортированных по минимальной цене.
3. «Книга» и «Книжный магазин» связаны соотношением многие-ко-многим. Выведите список всех связанных книг и магазинов, отсортированных по книгам, сортировка по магазинам произвольная.

Текст программы:

```
from operator import itemgetter


class Book:
    """Книга"""

    def __init__(self, id, price, name, shop_id):
        self.id = id
        self.price = price
        self.name = name
        self.shop_id = shop_id

class Shop:
    """Книжный магазин"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class BookShop:
    """ 'Книги магазина' для реализации связи многие-ко-многим """
    def __init__(self, shop_id, book_id):
        self.shop_id = shop_id
        self.book_id = book_id
```

```
Shops = [
    Shop(1, 'Книжный лабиринт'),
    Shop(2, 'Читай-город'),
    Shop(3, 'Антикварная лавка'),
    Shop(4, 'Достоевский'),
    Shop(5, 'Аллегория')
]

Books = [
    Book(1, 100, 'Анжелика', 1),
    Book(2, 339, 'Левша', 2),
    Book(3, 254, 'Повесть временных лет', 2),
    Book(4, 468, 'Курс Python', 3),
    Book(5, 179, 'Анна Каренина', 3),
    Book(6, 700, 'Всадник без головы', 4),
    Book(7, 660, 'Евгений Онегин', 5)
]

Books_Shops = [
    BookShop(1, 1),
    BookShop(2, 2),
    BookShop(2, 3),
    BookShop(3, 4),
    BookShop(3, 5),
    BookShop(4, 6),
    BookShop(5, 7)
]


def main():
    one_to_many = [(b.name, b.price, s.name)
                   for s in Shops
                   for b in Books
                   if b.shop_id == s.id]

    many_to_many_temp = [(s.name, bs.shop_id, bs.book_id)
                         for s in Shops
                         for bs in Books_Shops
                         if s.id == bs.shop_id]

    many_to_many = [(name, b.name)
                    for name, shop_id, book_id in many_to_many_temp
                    for b in Books if b.id == book_id]

    print('Задание B1')
    res_1 = []
    for i in Shops:
        book = list(filter(lambda a: a[2] == i.name, one_to_many))
        if i.name[0] == 'A':
            s_name = [x for x, _, _ in book]
            res_1.append((i.name, s_name))
    print(res_1)

    print('Задание B2')
    res unsorted = []
    for s in Shops:
        s_book = list(filter(lambda i: i[2] == s.name, one_to_many))
        if len(s_book) > 0:
            prices = [price for _, price, _ in s_book]
            prices_min = min(prices)
            res_unsorted.append((s.name, prices_min))
    res_2 = sorted(res_unsorted, key=itemgetter(1))
    print(res_2)
```

```
    print('Задание B3')
    res_13 = {}
    # Перебираем все отделы
    many_to_many.sort(key=lambda i: i[1])
    print(many_to_many)

if __name__ == '__main__':
    main()
```

Результаты выполнения:

```
Задание B1
[('Антикварная лавка', ['Курс Python', 'Анна Каренина']), ('Аллегория', ['Евгений Онегин'])]
Задание B2
[('Книжный лабиринт', 100), ('Антикварная лавка', 179), ('Читай-город', 254), ('Аллегория', 660), ('Достоевский', 700)]
Задание B3
[('Книжный лабиринт', 'Анжелика'), ('Антикварная лавка', 'Анна Каренина'), ('Достоевский', 'Всадник без головы'), ('Аллегория', 'Евгений Онегин'), ('Антикварная лавка', 'Курс Pyt

Process finished with exit code 0
```

```
('Антикварная лавка', 'Курс Python'), ('Читай-город', 'Левша'), ('Читай-город', 'Повесть временных лет')]
```

(продолжение результата на третий запрос)